LEVEL II (12)

AD A109445

DTNSRDC-81/083

DAVID W. TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER

Bethesda, Maryland 20084

THE EVEN-RHO AND EVEN-EPSILON ALGORITHMS FOR ACCELERATING CONVERGENCE OF A NUMERICAL SEQUENCE

by

Robert P. Eddy

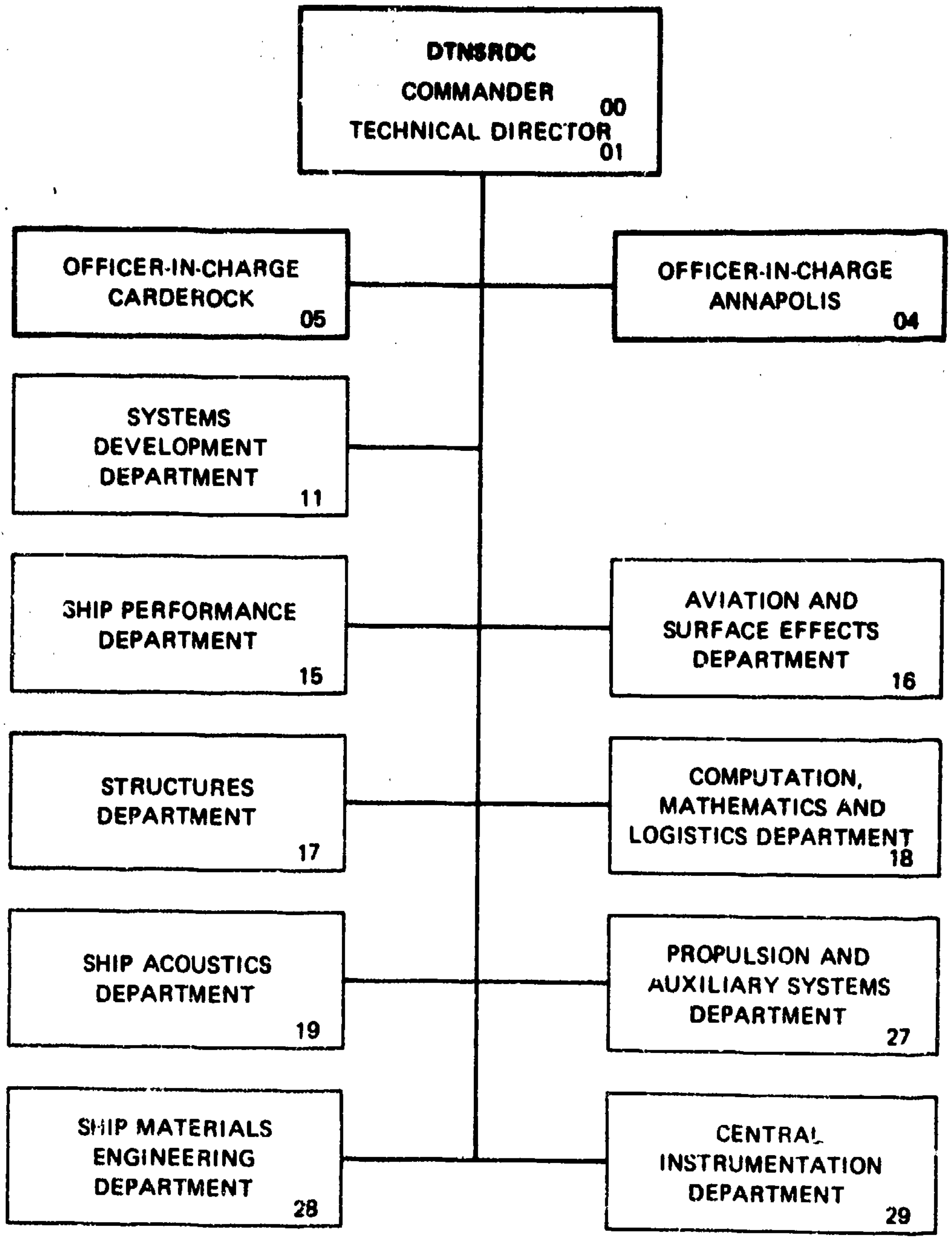
COMPUTATION, MATHEMATICS AND LOGISTICS DEPARTMENT
RESEARCH AND DEVELOPMENT REPORT

December 1981 DTNSRDC-81/083

THE EVEN-RHO AND EVEN-EPSILON ALGORITHMS FOR ACCELERATING CONVERGENCE OF A NUMERICAL SEQUENCE

82 01 08 141

MAJOR DTNSRDC ORGANIZATIONAL COMPONENTS


DTNSRDC
COMMANDER 00
TECHNICAL DIRECTOR 01
OFFICER-IN-CHARGE CARDEROCK 05
OFFICER-IN-CHARGE ANNAPOLIS 04
SYSTEMS DEVELOPMENT DEPARTMENT 11
SHIP PERFORMANCE DEPARTMENT 15
AVIATION AND SURFACE EFFECTS DEPARTMENT 16
STRUCTURES DEPARTMENT 17
COMPUTATION, MATHEMATICS AND LOGISTICS DEPARTMENT 18
SHIP ACOUSTICS DEPARTMENT 19
PROPULSION AND AUXILIARY SYSTEMS DEPARTMENT 27
SHIP MATERIALS ENGINEERING DEPARTMENT 28
CENTRAL INSTRUMENTATION DEPARTMENT 29

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER DTNSRDC-81/083	2. GOVT ACCESSION NO. AD-A109 445	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) THE EVEN-RHO AND EVEN-EPSILON ALGORITHMS FOR ACCELERATING CONVERGENCE OF A NUMERICAL SEQUENCE		5. TYPE OF REPORT & PERIOD COVERED Final Report 1976-1980
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Robert P. Eddy		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS David W. Taylor Naval Ship Research and Development Center Bethesda, Maryland 20084		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS (See reverse side)
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE December 1981
		13. NUMBER OF PAGES 60
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Acceleration of Convergence
Extrapolation of Sequences

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Two closely related algorithms are presented for extrapolating to the limit of a scalar sequence. One, the even-epsilon algorithm, is due to Wynn; it permits systematic calculation of the array of Shank's transforms or, equivalently, of the related Padé Table. The other, the even-rho algorithm, is closely related to the first and is also based on Wynn's work; however, it has different properties and has not enjoyed the same theoretical

(Continued on reverse side)

(Block 10)

Math Sciences Research Programs
Task Area SR-0140301 (Math Sciences) NAVSEA

(Block 20 continued)

development. Singular rules and near-singular rules are developed for both algorithms to handle situations in which adjacent tabular entries are equal or nearly equal, leading to zero or very small divisors. Computer programs implementing these algorithms are given along with sample output. An appreciable amount of historical background material is included.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Two closely related algorithms are presented for extrapolating to the limit of a scalar sequence. One, the even-epsilon algorithm, is due to Wynn; it permits systematic calculation of the array of Shank's transforms or, equivalently, of the related Padé Table. The other, the even-rho algorithm, is closely related to the first and is also based on Wynn's work; however, it has different properties and has not enjoyed the same theoretical development. Singular rules and near-singular rules are developed for both algorithms to handle situations in which adjacent tabular entries are equal or nearly equal, leading to zero or very small divisors. Computer programs implementing these algorithms are given along with sample output. An appreciable amount of historical background material is included.

ADMINISTRATIVE INFORMATION

This research was carried out under the Mathematical Sciences Research Program, Task Area SR-0140301 (Math Sciences), supported by the Naval Sea Systems Command.

1. INTRODUCTION

1.1 PURPOSE AND SCOPE

The purpose of this report is to present two computational algorithms, called the even-rho and the even-epsilon algorithms, for accelerating convergence of numerical sequences and, in addition, to present singular rules and near singular rules for handling cases where zero or near zero divisors appear in the computations. Numerical sequences arise in a wide variety of contexts in applied mathematics and engineering; examples are the successive partial sums of an infinite series and the successive outputs of an iterative process $s_{n+1} = f(s_n)$. Singular cases require special treatment to avoid excessive loss of significance. The computational algorithms as given here are new but their derivation depends heavily on the work of Peter Wynn. The singular rules and near singular rules appear here for the first time.

The even-epsilon algorithm is a computational adaptation of a formula given by Wynn (1966)[1*] relating adjacent members of the Padé table; this table is a rectangular array of rational functions the series expansions of which agree in a specified manner with a given series.

The even-rho algorithm is based on developments closely parallel to the above but stemming from the reciprocal differences introduced by Thiele (1909)[2] to facilitate interpolation by means of rational functions. This algorithm makes its debut here.

These two algorithms, although very similar, have different properties and behave differently when applied to any particular numerical sequence. Five examples of numerical sequences and the results of applying the even-rho and even-epsilon algorithms to them are given in the appendixes. Computer programs are also given.

An effort has been made to include sufficient historical background to provide perspective for the even-rho and even-epsilon algorithms, i.e., to show where they fit into the scheme of efforts to accelerate convergence of numerical sequences. To a limited extent this historical material also provides background for a companion report, Eddy (1980)[3] on a very successful method of accelerating convergence of a vector sequence generated in the iterative solution of a system of linear algebraic equations.

1.2 SEQUENCES AND SERIES

Numerical sequences and series which must be evaluated numerically arise in many contexts in applied mathematics. A sequence may be the successive outputs of an iterative process

$$s_n = f(s_{n-1}) \qquad (1.2\text{-}1)$$

or it may represent the successive partial sums of a power series

$$s_n = \sum_{j=0}^{n} a_j t^j \qquad (1.2\text{-}2)$$

*A complete listing of references is given on page 53.

Conversely, with any numerical sequence s_0, s_1, s_2,... there can be associated a power series

$$S(t) = s_0 + \sum_{j=1}^{\infty} (s_j - s_{j-1}) t^j \qquad (1.2\text{-}3)$$

having partial sums

$$S_n(t) = s_0 + \sum_{j=1}^{n} (s_j - s_{j-1}) t^j \qquad (1.2\text{-}4)$$

which, for $t = 1$, collapse to the original sequence:

$$S_n(1) = s_n \qquad (1.2\text{-}5)$$

Such a sequence or series may converge with sufficient rapidity to be computationally useful as it stands; it may converge so slowly that acceleration techniques are required; or it may diverge so that special techniques are required to determine the antilimit from which it is diverging.

1.3 CLASSICAL BACKGROUND

Ever since infinite series came into use about the time of Newton, the problem of accelerating the convergence of slowly converging series, or of assigning a value to a divergent series, had to be dealt with. Various methods for doing so, developed during the 18th and 19th centuries, are discussed in, for instance, Knopp's classic treatise, Knopp (1951),[4] especially chapters 8, 13, and 14, and also in Kline (1972)[5], chapters 20 and 47. Among the best known of these methods is the Euler transformation

$$\sum_{n=0}^{\infty} (-1)^n a_n = \sum_{n=0}^{\infty} (-1)^n (\Delta^n a_o)/2^{n+1} \qquad (1.3\text{-}1)$$

where $\Delta^n a_0$ is the nth forward difference of the a's beginning at a_0. Another well-known method which is still in use is the Euler-Maclaurin summation formula

$$\sum_{n=0}^{N} f(n) = \int_0^N f(x)dx - \frac{1}{2}(f(N)-f(0)) + \sum_{n=1}^{\infty} \frac{B_{2n}}{(2n)!}(f(N)^{(2N-1)}-f(0)^{(2N-1)}) \quad (1.3\text{-}2)$$

where the B_{2n} are the Bernoulli numbers.

Toward the end of the 19th century two new methods of dealing with the convergence problem were developed. The first, called summability theory and associated primarily with the names of Cesåro and Hölder, employed various "means" which were linear combinations of successive partial sums. These means proved to be of great theoretical value but seem to be not very useful computationally. The second new method involved the use of rational functions, either as continued fractions or as the ratio of two polynomials; it has largely supplanted other methods for accelerating convergence.

The way was opened by Stieltjes who, in a series of papers beginning in 1889, exploited the idea of converting the infinite tail of a series (the part left after removing a partial sum) into a continued fraction. Since the domain of convergence of a continued fraction is quite different from that of a power series, this technique often yields useful reults. See Wall (1948)[6].

Soon thereafter came Padé who, in his 1892 thesis, studied the relations between a given power series and the rectangular array of rational functions (later known as the Padé table) related to it as follows: the rational function in the pth row and qth column has numerator of degree p and denominator of degree q and its power series expansion agrees with the given series through the term of degree p + q. (Actually, the array treated by Padé was the transpose of this one, but this description conforms to current usage.)

To evaluate the limit of convergent series, or the antilimit of a divergent series, one evaluates the array of functions in the Padé table and looks for convergence down the columns, q = constant, or along the main diagonal, p = q.

For purely numerical work, the most effective way of accomplishing this evaluation is with the epsilon algorithm (or the even-epsilon algorithm) described in Section 3.

Considerable theoretical work has been done on Padé approximation during the past two decades; the present state of the art is summarized in Baker (1975).[7] Earlier work is described in the standard works on continued fractions: Perron (1929)[8] and Wall (1948).[6]

By far the best known and most widely used example of summation by approximation by a rational function is the simplest possible case: the rational function is merely the usual expression for the sum of a geometric series with the common ratio determined from three successive partial sums. This simple function has been discovered and used by various authors, most notably Aitken (1926)[9] and Shanks (1949)[10] and (1955).[11] Because of the second difference in the denominator, Aitken called it the delta squared process. This extrapolation process can be written in any of the following equivalent forms:

$$A_n = s_{n-1} - (s_n - s_{n-1})^2/(s_{n+1} - 2s_n + s_{n-1})$$

$$= s_n - (s_{n+1} - s_n)(s_n - s_{n-1})/(s_{n+1} - 2s_n + s_{n-1})$$

$$= s_{n+1} - (s_{n+1} - s_n)^2/(s_{n+1} - 2s_n + s_{n-1})$$

$$= \frac{s_{n+1}\, s_{n-1} - s_n^2}{s_{n+1} - 2s_n + s_{n-1}} \qquad (1.3\text{-}3)$$

$$\frac{1}{A_n - s_n} = \frac{1}{s_{n+1} - s_n} - \frac{1}{s_n - s_{n-1}} \qquad (1.3\text{-}4)$$

Expression (1.2-3) is the form usually given; (1.2-4) is displayed because it is a special case of the even-epsilon algorithm to be described in Section 3.

For a comprehensive but concise account of acceleration methods currently considered to be of interest, especially the epsilon algorithm, see the lecture notes of Claude Brezinski (1977)[12] and also his textbook, Brezinski (1978).[13]

2. THE RHO AND EVEN-RHO ALGORITHMS

2.1 THE RHO ALGORITHM

The rho algorithm was so named by Peter Wynn who, in Wynn (1956),[14] the first of two important papers published a few months apart, called attention to the usefulness of reciprocal differences for summary series and for extrapolating to the limit of sequences.

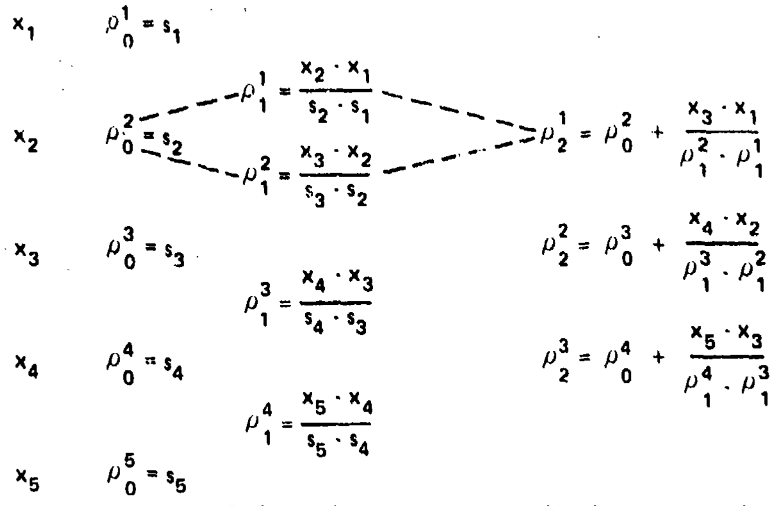
Reciprocal differences, denoted by ρ, had been invented a half century earlier by Thiele (1909)[2] to facilitate interpolation by rational functions. The interpolating function appears as a continued fraction which is the analog of Newton's polynomial interpolation formula based on his divided differences. Analytical properties [illegible] hese reciprocal differences were investigated in the ensuing years by Nörlund whose famous text, Nörlund (1924),[15] remains the definitive treatment of the subject. A similar treatment is found in Milne-Thomson (1933).[16]

The reciprocal differences corresponding to a given set of arguments $\{X_n\}$ and function values $\{s_n\}$ are defined recursively:

$$\rho_m^n = \rho_{m-2}^{n+1} + \frac{X_{n+m} - X_n}{\rho_{m-1}^{n+1} - \rho_{m-1}^{n}} \qquad (2.1\text{-}1)$$

$$\rho_0^n = s_n, \qquad \rho_{-1}^n \equiv 0 \qquad (2.1\text{-}2)$$

They are customarily displayed in the rho array as shown in Figure 2.1.



x_1 $\rho_0^1 = s_1$
$\rho_1^1 = \frac{x_2 - x_1}{s_2 - s_1}$
x_2 $\rho_0^2 = s_2$
$\rho_2^1 = \rho_0^2 + \frac{x_3 - x_1}{\rho_1^2 - \rho_1^1}$
$\rho_1^2 = \frac{x_3 - x_2}{s_3 - s_2}$
x_3 $\rho_0^3 = s_3$
$\rho_2^2 = \rho_0^3 + \frac{x_4 - x_2}{\rho_1^3 - \rho_1^2}$
$\rho_1^3 = \frac{x_4 - x_3}{s_4 - s_3}$
x_4 $\rho_0^4 = s_4$
$\rho_2^3 = \rho_0^4 + \frac{x_5 - x_3}{\rho_1^4 - \rho_1^3}$
$\rho_1^4 = \frac{x_5 - x_4}{s_5 - s_4}$
x_5 $\rho_0^5 = s_5$

Figure 2.1 - The Rho Array (Old Notation)

It will be seen that any particular ρ_m^n depends explicitly only on three previously calculated rho's lying at the other three vertices of a rhombus. However, implicitly ρ_m^n depends on all m+1 of the pairs (X_n, s_n), $(X_{n+1}, s_{n+1})\ldots(X_{n+m}, s_{n+m})$.

Systematic computation of the rho array can proceed according to either of two patterns:

- columnwise - all of the input sequence $\rho_o^n = s_n$, then all of the first column ρ_1^n, etc., or
- by upsloping diagonals - as each new $\rho_o^n = s_n$ is obtained, calculate in turn ρ_1^{n-1}, ρ_{n-1}^1. This latter pattern has two distinct advantages,
- it reduces storage requirements since calculation along any upsweeping diagonal requires input from only itself and the immediately preceding upsweeping diagonal, and
- convergence can be monitored in all columns involved on each sweep so that the basic input interation can be halted as soon as possible.

The property of reciprocal differences which makes them so useful for extrapolating to the limit of a sequence (or, equivalently, for summing a series) is that, if the general term of a sequence is given by a rotional function of the term index (ordinarily X_n=n) in the form

$$s_n = \frac{a_k X_n^k + a_{k-1} X_n^{k-1} + \ldots + a_1 X_n + a_0}{X_n^k + b_{k-1} X_n^{k-1} + \ldots + b_1 X_n + b_0} \qquad (2.1\text{-}3)$$

then

$$\rho_{2k}^n = a_k \quad \text{for } n = 1, 2, 3, \ldots \qquad (2.1\text{-}4)$$

Since obviously

$$\lim_{X_n \to \infty} s_n = a_k \qquad (2.1\text{-}5)$$

it is clear that calculation of the even ordered reciprocal differences provides a systematic way of finding the limit or antilimit of such a sequence. If relation

(2.1-3) is not exact, but is an increasingly good approximation as n gets larger, then there will be convergence in the 2kth column of the rho array.

The rho algorithm, in contrast to the epsilon algorithm discussed in Section 3, seems to have received very little attention in the literature since the publication of Wynn (1956)[14] and convergence theorems for it are lacking. It has, however, been touched upon several times by Brezinski (1971),[17] (1972),[18] and especially (1977).[12]

2.2 THE EVEN-RHO ALGORITHM

Since only the even-ordered reciprocal differences are useful for the summation/extrapolation process, it would be desirable to develop a recursion relation involving them alone. This indeed can be done and the result is Equation (2.2-4) which, to the best of the author's knowledge, appears here for the first time in print. The novelty is, however, trivial since the pattern of elimination which leads to this recursion relation is precisely the one employed by Wynn (1966)[1] to develop the even-epsilon algorithm from his earlier epsilon algorithm. In order to make this report more self contained, the derivation is given here for the even-rho algorithm.

Consider a rhombus shaped portion of the rho array as shown in Figure 2.2.

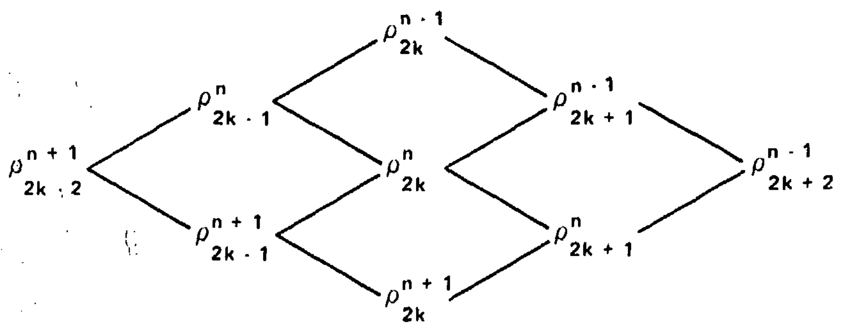


Figure 2.2 - A Portion of the Rho Array

The rho recursion relation (2.1-1), now for convenience rewritten

$$\rho^{n}_{2k+1} = \rho^{n+1}_{2k-1} + \frac{X_{2k+n+1} - X_n}{\rho^{n+1}_{2k} - \rho^{n}_{2k}} \tag{2.2-1}$$

gives

$$\rho^{n}_{2k+1} - \rho^{n+1}_{2k-1} = (X_{2k+n+1} - X_n)/(\rho^{n+1}_{2k} - \rho^{n}_{2k})$$

and, with n-1 replacing n,

$$\rho_{2k+1}^{n-1} - \rho_{2k-1}^{n} = (X_{2k+n}-X_{n-1})/(\rho_{2k}^{n}-\rho_{2k}^{n-1})$$

Subtraction results in

$$(\rho_{2k+1}^{n}-\rho_{2k+1}^{n-1}) - (\rho_{2k-1}^{n+1}-\rho_{2k-1}^{n}) = \frac{X_{2k+n+1} - X_{n}}{\rho_{2k}^{n+1} - \rho_{2k}^{n}} - \frac{X_{2k+n} - X_{n-1}}{\rho_{2k}^{n} - \rho_{2k}^{n-1}}$$

Now Equation (2.2-1) can be used to replace the ordinary difference on the left with reciprocal differences:

$$\frac{X_{2k+n+1} - X_{n-1}}{\rho_{2k+2}^{n-1} - \rho_{2k}^{n}} - \frac{X_{2k+n} - X_{n}}{\rho_{2k}^{n} - \rho_{2k-2}^{n+1}} = \frac{X_{2k+n+1} - X_{n}}{\rho_{2k}^{n+1} - \rho_{2k}^{n}} - \frac{X_{2k+n} - X_{n-1}}{\rho_{2k}^{n} - \rho_{2k}^{n-1}} \qquad (2.2\text{-}2)$$

Note that on the left side the differences lie in a horizontal direction, whereas on the right side they lie in a vertical direction.

For ordinary use in summation of series or extrapolation of sequences it is natural to take

$$X_{n} = n \quad (n = 1, 2, 3, \ldots) \qquad (2.2\text{-}3)$$

This substitution yields on the even-rho algorithm

$$\frac{2k+2}{\rho_{2k+2}^{n-1} - \rho_{2k}^{n}} - \frac{2k}{\rho_{2k}^{n} - \rho_{2k-2}^{n+1}} = \frac{2k+1}{\rho_{2k}^{n+1} - \rho_{2k}^{n}} - \frac{2k+1}{\rho_{2k}^{n} - \rho_{2k}^{n-1}} \qquad (2.2\text{-}4)$$

$$\rho_{0}^{n} = s_{n} \quad (n = 1, 2, 3, \ldots) \quad (k = 0, 1, 2, 3, \ldots)$$

It is not necessary to specify the ρ_{-2}^{n} since the only term in which they would appear has k=0 in the numerator.

There is now a cross pattern, shown in Figure 2.3, which corresponds to the rhombus pattern associated with the ordinary rho algorithm.

$$\begin{tikzcd}
 & \rho^{n-1}_{2k} \arrow[d, dash] & \\
\rho^{n+1}_{2(k-1)} \arrow[r, dash] & \rho^{n}_{2k} \arrow[r, dash] \arrow[d, dash] & \rho^{n-1}_{2(k+1)} \\
 & \rho^{n+1}_{2k} &
\end{tikzcd}$$

Figure 2.3 - The Even-Rho Pattern (Old Notation)

2.3 NEW NOTATION

The notation ρ^{n}_{2k} used so far has corresponded to that used by Wynn in his later papers on the epsilon algorithm and also by Brezinski. The lower index, m or 2k, denotes the column, and the upper index, n, denotes the downsloping diagonal on which it lies. The latter thus emphasizes the first member of the input sequence, s_n, which enters into the calculation of ρ^{n}_{2k} (see Figure 2.1).

However, for a computational algorithm which sweeps through the array along upsloping diagonals, it is much more convenient to have the upper index denote the <u>upsloping</u> diagonal. Hence it is constant along each sweep, and thus emphasizes the last member of the input sequence, s_p, which enters into the current calculation. Henceforth the upper index denoting the upsweeping diagonal will be p instead of n; they are related by

$$p = n + 2k \tag{2.3-1}$$

In this notation the even-rho array appears as in Figure 2.4.

The even-rho recursion, relating the five elements lying on a cross pattern, appears as in Figure 2.5.

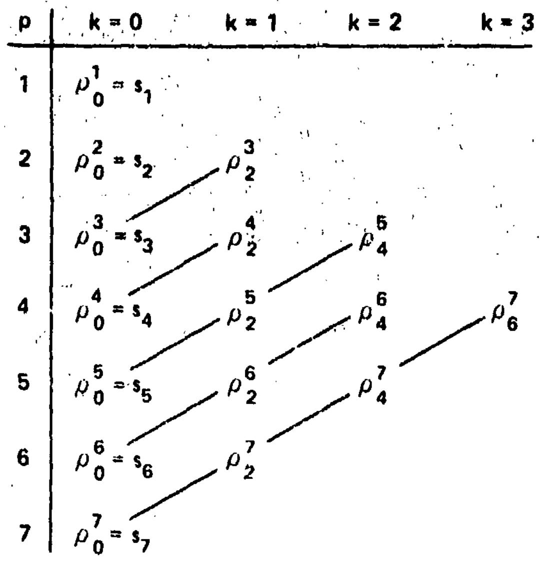


Figure 2.4 - The Even-Rho Array (New Notation)

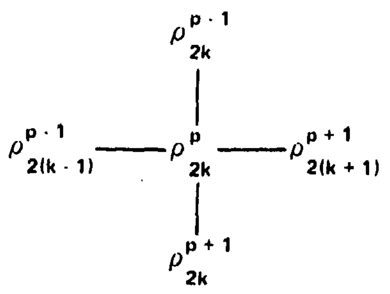


Figure 2.5 - The Even-Rho Pattern (New Notation)

Equation (2.2-4) becomes

$$\frac{2(k+1)}{\rho^{p+1}_{2(k+1)} - \rho^{p}_{2k}} - \frac{2k}{\rho^{p}_{2k} - \rho^{p-1}_{2(k-1)}} = \frac{2k+1}{\rho^{p+1}_{2k} - \rho^{p}_{2k}} - \frac{2k+1}{\rho^{p}_{2k} - \rho^{p-1}_{2k}} \tag{2.3-2}$$

$$(p = 2, 3, 4, \ldots; k = 0, 1, 2, \ldots [(p-1)/2])$$

$$\rho^{p}_{o} = s_{p} \quad (p = 1, 2, 3, \ldots)$$

The new notation has an additional advantage--mainly esthetic, to be sure--in that the upper indices in Equation (2.3-2) behave properly instead of running backward in the horizontal differences as they did in the old notation of Equation (2.2-4).

2.4 COMPUTATIONAL ALGORITHM

The two terms on the left side of Equation (2.3-2) have the same structure, that of principal parts of reciprocal differences taken in a horizontal direction. Likewise the two terms on the right are principal parts of reciprocal differences taken in a vertical direction. Each such term is used twice, once on each of two successive sweeps along upsloping diagonals. It is therefore advantageous to define the auxiliary quantities

$$H^{p}_{k} = \frac{2k}{\rho^{p}_{2k} - \rho^{p-1}_{2(k-1)}} \tag{2.4-1}$$

$$V^{p}_{k} = \frac{2k+1}{\rho^{p}_{2k} - \rho^{p-1}_{2k}} \tag{2.4-2}$$

In terms of these quantities, the equations for calculating the ρ^{p}_{2k} along an upsloping diagonal indexed by p, as in Figure 2.4, become

$$V_k^p = \frac{2k+1}{\rho_{2k}^p - \rho_{2k}^{p-1}} \quad (p = 2, 3, 5, \ldots) \tag{2.4-3}$$

$$H_{k+1}^p = H_k^{p-1} + V_k^p - V_k^{p-1} \quad (p = 3, 4, 5, \ldots) \tag{2.4-4}$$

$$\rho_{2(k+1)}^p = \rho_{2k}^{p-1} + \frac{2(k-1)}{H_{k+1}^p} \quad (p = 3, 4, 5, \ldots) \tag{2.4-5}$$

$$(K = 0, 1, 2, \ldots [(p-1)/2], \;\; p = 2, 3, 4, \ldots) \tag{2.4-6}$$

with initial conditions

$$\left.\begin{array}{l} \rho_o^p = s_p \\ H_o^p = 0 \end{array}\right\} \quad (p = 1, 2, 3, \ldots) \tag{2.4-7}$$

Since calculations along any upsloping diagonal, indexed by p, require as input only previously computed values for the same p and also for p-1, it is necessary to maintain six arrays: three for the currently computed values of V, H, and rho, and three similar arrays from the previous (p-1) sweep. Their maximum length, L, is related to P, the maximum value of p, by

$$L = [(P+1)/2]$$

where [x] means the greatest integer, n, satisfying $n \leq x < n+1$.

3. THE EPSILON AND EVEN-EPSILON ALGORITHMS

3.1 SHANKS' TRANSFORMS AND WYNN'S EPSILON ALGORITHM

The epsilon algorithm was so named by Peter Wynn who introduced it in Wynn (1956)[19] as a practical method of calculating the array of transforms $e_m(S_n)$

discussed by Shanks in his thesis, Shanks (1955).[11] The latter is credited by Gragg (1972),[20] page 2, as being one of two principal sources of stimulus for modern interest in Padé approximation and related topics. Actually, an earlier version, Shanks (1949),[10] had attained unusually wide circulation for an internal memorandum and was widely known among numerical analysts in the early 1950's.

Shank's considered sequences of the form

$$s_n = a_o + \sum_{k=1}^{m} a_k r_k^n \qquad (3.1\text{-}1)$$

which arose, for example, in studying the decay of a mixture of radioactive substances. If all r_k are distinct and satisfy

$$|r_k| < 1 \qquad (3.1\text{-}2)$$

then

$$\lim_{n\to\infty} s_n = a_o \qquad (3.1\text{-}3)$$

so that a_o is the quantity of interest in this context.

Two properties of the sequence (3.1-1) should be noted:

(1) each s_n is the sum of the nth partial sums of m different geometric series;

(2) $s_n - a_o = \sum_{k=1}^{m} a_k r_k^n$ has the form of the general solution of a homogenous linear finite difference equation with constant coefficients

$$\sum_{j=0}^{m} C_j (s_{n+j} - a_o) = 0 \quad (n = 1, 2, 3, \ldots) \qquad (3.1\text{-}4)$$

where the r_k are the roots of the characteristic equation

$$\sum_{j=0}^{m} c_j r^j = 0 \tag{3.1-5}$$

The auxiliary condition

$$\sum_{j=0}^{m} c_j \neq 0 \tag{3.1-6}$$

is required to exclude the singular case in which one of the roots is unity.

Either Equation (3.1-1) or Equation (3.1-4) can be regarded as characterizing the sequences to which the theory of this chapter applies.

Shanks gave an array of transforms

$$e_m(s_n) = \frac{\begin{vmatrix} s_n & s_{n+1} & \cdots & s_{n+m} \\ s_{n+1} - s_n & s_{n+2} - s_{n+1} & \cdots & s_{n+m+1} - s_{n+m} \\ \vdots & \vdots & & \vdots \\ s_{n+m} - s_{n+m-1} & \cdot \quad \cdot & \cdot & s_{n+2m} - s_{n+2m-1} \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ s_{n+1} - s_n & s_{n+2} - s_{n+1} & \cdots & s_{n+m+1} - s_{n+m} \\ \vdots & \vdots & & \vdots \\ s_{n+m} - s_{n+m-1} & \cdots & & s_{n+2m} - s_{n+2m-1} \end{vmatrix}} \tag{3.1-7}$$

for estimating a_o, the limit (or antilimit) of the sequence in terms of 2m+1 successive elements, s_n, s_{n+1}, ... s_{n+2m}. Here, n denotes the starting point in the sequence and m denotes the order of the approximation as in Equation (3.1-1). In particular, for m=1 this yields the well known Aitken delta-squared extrapolation formula already shown in Equation (1.3-3).

The expressions (3.1-7) given by Shanks, the ratio of two determinants of Hankel type, become practically useless for direct computation even when m is still a small integer, e.g., m=4. This difficulty was soon surmounted by Wynn (1956)[19] who showed that, with the aid of some new intermediate quantities, a very simple recursion relation holds.

He set

$$\left.\begin{aligned} \epsilon_{2m}^{n} &= e_m(s_n) \\ \epsilon_{2m+1}^{n} &= 1/e_m(s_{n+1}-s_n) \end{aligned}\right\} \qquad (3.1\text{-}8)$$

and showed that they satisfy

$$\left.\begin{aligned} &\epsilon_{k+1}^{n} = \epsilon_{k-1}^{n+1} + \frac{1}{\epsilon_k^{n+1} - \epsilon_k^{n}} \\ &\qquad\qquad\qquad\qquad k=0,\ 1,\ 2,\ 3,\ldots \\ &\qquad\qquad\qquad\qquad n=0,\ 1,\ 2,\ 3,\ldots \\ &\epsilon_o^{n} = s_n \qquad \epsilon_{-1}^{n} = 0 \end{aligned}\right\} \qquad (3.1\text{-}9)$$

This is the epsilon algorithm; it is very similar to the rho algorithm, Equation (2.1-1). Correspondingly, there is the epsilon array shown in Figure 3.1.

$$\begin{tikzcd}
n & k=0 & k=1 & k=2 \\
1 & \epsilon_0^1 = s_1 & & \\
 & & \epsilon_1^1 = \frac{1}{s_2 - s_1} \arrow[dr, dashed] & \\
2 & \epsilon_0^2 = s_2 \arrow[ur, dashed] \arrow[dr, dashed] & & \epsilon_2^1 = \epsilon_0^2 + \frac{1}{\epsilon_1^2 - \epsilon_1^1} \\
 & & \epsilon_1^2 = \frac{1}{s_3 - s_2} \arrow[ur, dashed] & \\
3 & \epsilon_0^3 = s_3 & & \epsilon_2^2 = \epsilon_0^3 + \frac{1}{\epsilon_1^3 - \epsilon_1^2} \\
 & & \epsilon_1^3 = \frac{1}{s_4 - s_3} & \\
4 & \epsilon_0^4 = s_4 & &
\end{tikzcd}$$

Figure 3.1 - The Epsilon Array (Old Notation)

Systematic computation of this array can proceed either columnwise or along up-sloping diagonals as was discussed for the rho algorithm in Section 2.1.

The relationship between the array of transforms (3.1-7) and the rational functions of the Padé table was pointed out in Shanks (1949)[10] and (1955);[11] the same relationship, but in the language of the epsilon array, was treated in Wynn (1961).[21] Specifically, let s_n be the nth partial sum of a power series

$$s_n = \sum_{j=0}^{n} c_j z^j \tag{3.1-10}$$

Substitute Equation (3.1-10) into the expression (3.1-7) for the $e_m(s_n)$ transform; then in both numerator and denominator multiply the first column by z^m, the second column by z^{m-1}, ... the last column by z^0; cancel out the powers of z in corresponding rows of numerator and denominator. What is left is very clearly the ratio of a polynomial of degree (n+m) to a polynomial of degree m:

$$e_m(s_n) = \frac{\begin{vmatrix} z^m s_n & z^{m-1} s_{n+1} & \cdot & \cdot & \cdot & z^0 s_{n+m} \\ c_{n+1} & c_{n+2} & \cdot & \cdot & \cdot & c_{n+m} \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ c_{n+m} & c_{n+m+1} & \cdot & \cdot & \cdot & c_{n+2m} \end{vmatrix}}{\begin{vmatrix} z^m & z^{m-1} & \cdot & \cdot & \cdot & z^0 \\ c_{n+1} & c_{n+2} & \cdot & \cdot & \cdot & c_{n+m} \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ \cdot & \cdot & & & & \cdot \\ c_{n+m} & c_{n+m+1} & \cdot & \cdot & \cdot & c_{n+2m} \end{vmatrix}} \qquad (3.1\text{-}11)$$

The converse of this result was given by the present author in 1951. In unpublished working notes, he showed that a systematic construction of rational functions, whose power series expansions agree as far as possible with a given power series, leads to the $e_m(s_n)$ transform as given in Equation (3.1-11).

3.2 THE EVEN-EPSILON ALGORITHM

As with the rho algorithm, since only even-ordered epsilons are useful for extrapolation, it would seem desirable to eliminate the odd-ordered epsilons and arrive at a recursion relation involving only even-ordered epsilons. This elimination was accomplished by Wynn (1966).[1] The details will not be repeated here since they have already been given in Section 2.2 on the rho algorithm. Likewise, the arguments for changing to a new notation, already given for the rho algorithm in Section 2.3, will not be repeated here. Suffice it to show in Figure 3.2 the even-epsilon array in the new notation:

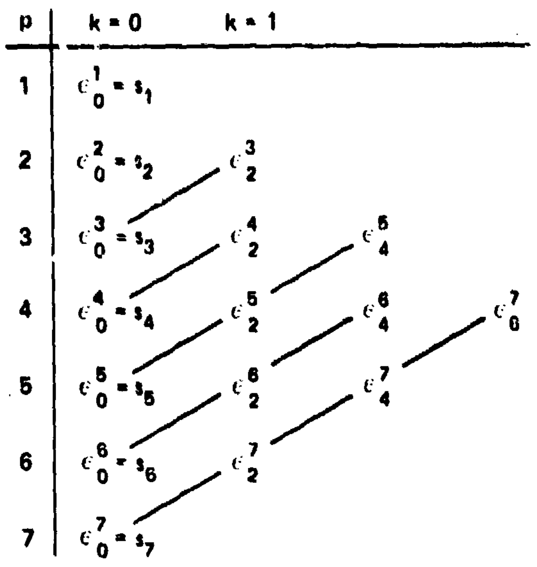

p | k = 0 | k = 1
1 | $\varepsilon_0^1 = s_1$
2 | $\varepsilon_0^2 = s_2$ | ε_2^3
3 | $\varepsilon_0^3 = s_3$ | ε_2^4 | ε_4^5
4 | $\varepsilon_0^4 = s_4$ | ε_2^5 | ε_4^6 | ε_6^7
5 | $\varepsilon_0^5 = s_5$ | ε_2^6 | ε_4^7
6 | $\varepsilon_0^6 = s_6$ | ε_2^7
7 | $\varepsilon_0^7 = s_7$

Figure 3.2 - The Even-Epsilon Array (New Notation)

Any five entries in this table, lying on a cross pattern as indicated in Figure 3.3,

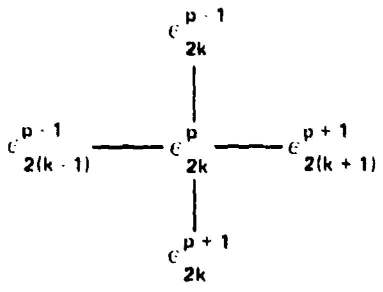


Figure 3.3 - The Even-Epsilon Pattern (New Notation)

are related according to Wynn's formula (Wynn (1966),[1] page 266)

$$\frac{1}{\varepsilon_{2(k+1)}^{p+1}-\varepsilon_{2k}^{p}} - \frac{1}{\varepsilon_{2k}^{p}-\varepsilon_{2(k-1)}^{p-1}} = \frac{1}{\varepsilon_{2k}^{p+1}-\varepsilon_{2k}^{p}} - \frac{1}{\varepsilon_{2k}^{p}-\varepsilon_{2k}^{p-1}} \qquad (3.2\text{-}1)$$

$$(p = 2, 3, 4, \ldots; \quad k=0, 1, 2, \ldots [(p-1)/2])$$

$$\epsilon_0^p = s_p, \quad \epsilon_{-2}^p = \infty \qquad (p = 1, 2, 3, \ldots)$$

This is the recursion relation among his transforms which Shanks came close to but never actually attained.*

Comparing this expression, Equation (3.2-1), with that for Aitken's delta-squared process (Shanks' e_1 process) shown in Equation (1.3-4), we see that a quantity E_k^p given by

$$\frac{1}{E_k^p - \epsilon_{2k}^p} = \frac{1}{\epsilon_{2k}^{p+1} - \epsilon_{2k}^p} - \frac{1}{\epsilon_{2k}^p - \epsilon_{2k}^{p-1}} \qquad (3.2\text{-}2)$$

is also given by

$$\frac{1}{E_k^p - \epsilon_{2k}^p} = \frac{1}{\epsilon_{2(k+1)}^{p+1} - \epsilon_{2k}^p} - \frac{1}{\epsilon_{2k}^p - \epsilon_{2(k-1)}^{p-1}} \qquad (3.2\text{-}3)$$

That is to say, at a given location in the even-epsilon array (Padé table), the same result is obtained by applying the delta-squared (e_1) transform either vertically or horizontally.

3.3 COMPUTATIONAL ALGORITHM

The two terms on the right side of Equation (3.2-1) have the same structure, that of principal parts of the next higher odd-ordered epsilons (see the epsilon recursion relation, (Equation (3.1-8)), the differences being taken in a vertical

*Oral communication from Shanks.

direction. A similar remark applies to the two terms on the left side, the differences being taken in a horizontal direction. This observation leads to the definitions

$$H_k^p = \frac{1}{\epsilon_{2k}^p - \epsilon_{2(k-1)}^{p-1}} \qquad (3.3\text{-}1)$$

$$V_k^p = \frac{1}{\epsilon_{2k}^p - \epsilon_{2k}^{p-1}} \qquad (3.3\text{-}2)$$

In terms of these quantities, the equations for calculating the ϵ_{2k}^p along an upsloping diagonal indexed by p, as in Figure 3.2, become

$$V_k^p = \frac{1}{\epsilon_{2k}^p - \epsilon_{2k}^{p-1}} \quad (p = 2, 3, 4, \ldots) \qquad (3.3\text{-}3)$$

$$H_{k+1}^p = H_k^{p-1} + V_k^p - V_k^{p-1} \quad (p = 3, 4, 5, \ldots) \qquad (3.3\text{-}4)$$

$$\epsilon_{2(k+1)}^p = \epsilon_{2k}^{p-1} + \frac{1}{H_{k+1}^p} \quad (p = 3, 4, 5, \ldots) \qquad (3.3\text{-}5)$$

$$(k=0,1,2,3,\ldots [(p-1)/2], \quad (p = 2, 3, 4, \ldots) \qquad (3.3\text{-}6)$$

with initial conditions

$$\left.\begin{aligned} \epsilon_0^p &= s_p \\ H_0^p &= 0 \end{aligned}\right\} \quad (p = 1, 2, 3, \ldots) \qquad (3.3\text{-}7)$$

Note that, for k=0, Equations (3.3-3) and (3.3-4) together yield the Aitken delta-squared transform, Equation (1.2-4).

As in the case of the even-rho algorithm, it is necessary to maintain two arrays for each of V, H, and epsilon, one each for the current values of V, H, and epsilon along the upsloping diagonal indexed by p, and one each for the last preceding diagonal indexed by p-1. Their maximum length, ℓ, is again related to P, the maximum value of p, by

$$L = [(P+1)/2] \tag{3.3-8}$$

where [x] means the greatest integer, n, satisfying $n \leq x < n+1$.

4. SINGULAR AND NEAR-SINGULAR RULES

4.1 NEED FOR SINGULAR AND NEAR-SINGULAR RULES

Computational algorithms, such as the even-rho and even-epsilon algorithms and also their ancestors the ordinary rho and epsilon algorithms, in which division plays a major role, are vulnerable to troubles arising from attempted division by zero (the singular case) or by a number which is very small in magnitude (the near-singular case). These situations are nuisances but need not stop the calculations; instead, one can use special formulas called singular rules for a zero division or near-singular rules for a near zero division.

The near-singular rules will be derived for the even-rho algorithm. Trivial modifications then yield the near-singular rules for the even-epsilon algorithm. In either case the singular rules are then an obvious limiting case of the near-singular rules.

Both singular and near-singular rules for his epsilon algorithm were given by Wynn (1962)[22] and (1963).[23] He also gave a related discussion for the even-epsilon algorithm in Wynn (1966).[1]

4.2 NEAR-SINGULAR RULES FOR THE EVEN-RHO ALGORITHM

A glance at the computational algorithm, Equations (2.4-3) - (2.4-7), reveals two places in which a small division may arise: the first is in the computation of V_k^p, the second in the computation of $\rho_{2(k+1)}^p$. In both cases the effects of near singularity are felt at several neighboring points encountered later ("down-stream") in the calculations. It is interesting to note, in both cases, instances wherein very large numbers cancel out by subtraction; they cancel exactly to within the accuracy of the approximations used. These are underlined in the formulas in which they occur.

Treatment is by a first order perturbation method: the small divisor, d, is assumed to have the following properties:

$$\left.\begin{array}{ll} (1) & |d| << 1 \\ (2) & 1/|d| >> \text{any other terms added to it} \\ (3) & d^2 \approx 0, \text{ i.e., terms in } d^2 \text{ are negligible} \end{array}\right\} \quad (4.2\text{-}1)$$

The simpler case will be discussed first. Suppose that, in the calculations associated with the index values p and k,

$$H^p_{k+1} = H^{p-1}_k + V^p_k - V^{p-1}_k = d \quad (4.2\text{-}2)$$

where d satisfies the criteria just given. Then, from Equation (2.4-5),

$$\rho^p_{2(k+1)} = \rho^{p-1}_{2k} + \frac{2k+2}{H^p_{k+1}} \approx \frac{2k+2}{d} \quad (4.2\text{-}3)$$

and from Equation (2.4-3)

$$V^p_{k+1} = \frac{2k+3}{\rho^p_{2(k+1)} - \rho^{p-1}_{2(k+1)}} \approx \left(\frac{2k+3}{2k+2}\right) d \quad (4.2\text{-}4)$$

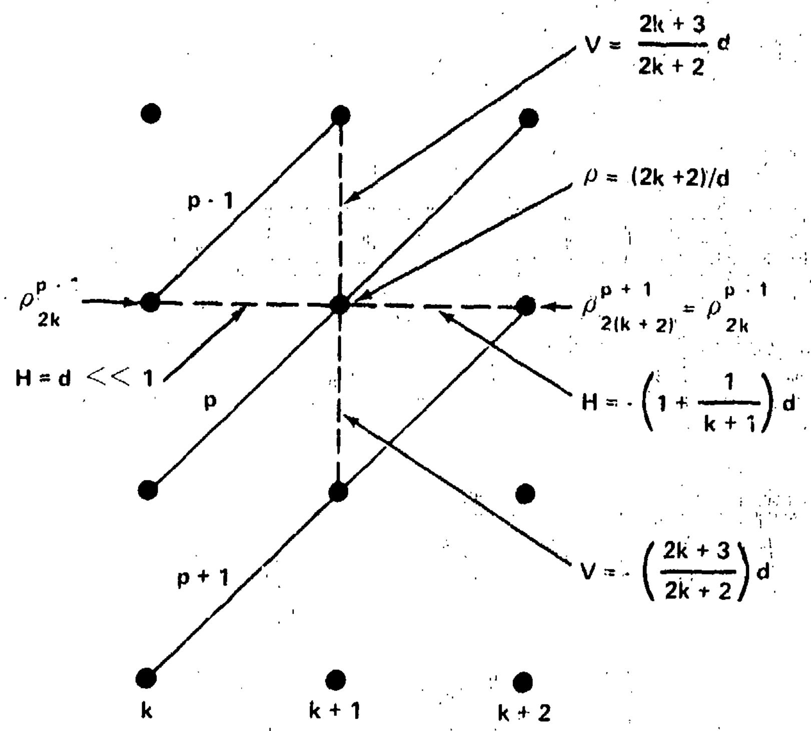
On the next following upsloping diagonal, indexed by p+1,

$$V^{p+1}_{k+1} = \frac{2k+3}{\rho^{p+1}_{2(k+1)} - \rho^p_{2(k+1)}} \approx - \left(\frac{2k+3}{2k+2}\right) d \quad (4.2\text{-}5)$$

$$H^{p+1}_{k+2} = H^p_{k+1} + V^{p+1}_{k+1} - V^p_{k+1} \approx d - 2 \left(\frac{2k+3}{2k+2}\right) d = - \left(\frac{2k+4}{2k+2}\right) d \quad (4.2\text{-}6)$$

$$\rho^{p+1}_{2(k+2)} = \rho^p_{2(k+1)} + \frac{2k+4}{H^{p+1}_{k+2}} \approx \rho^{p-1}_{2k} + \underline{\frac{2k+2}{d}} - \underline{\frac{(2k+4)\ (2k+2)}{(2k+4)d}} = \rho^{p-1}_{2k} + 0 \quad (4.2\text{-}7)$$

These results are summarized graphically in Figure 4.1.



$V = \frac{2k+3}{2k+2} d$
$\rho = (2k+2)/d$
ρ^{p-1}_{2k}
$\rho^{p+1}_{2(k+2)} = \rho^{p-1}_{2k}$
$H = d << 1$
$H = -\left(1 + \frac{1}{k+1}\right) d$
$V = -\left(\frac{2k+3}{2k+2}\right) d$
p - 1
p
p + 1
k
k + 1
k + 2

Figure 4.1 - First Case of Near-Singular Rules for the Even-Rho Algorithm

The corresponding singular rules are now trivial: for d = 0,

$$V^{p}_{k+1} = V^{p+1}_{k+1} = H^{p}_{k+1} = H^{p+1}_{k+2} = 0 \qquad (4.2\text{-}8)$$

$$\rho^{p}_{2(k+1)} = \infty \qquad (4.2\text{-}9)$$

$$\rho^{p+1}_{2(k+2)} = \rho^{p-1}_{2k} \qquad (4.2\text{-}10)$$

A more complicated situation arises if a newly calculated ρ^p_{2k} is almost equal to ρ^{p-1}_{2k} directly above it in the even-rho array:

$$\rho^p_{2k} - \rho^{p-1}_{2k} = d \qquad (4.2\text{-}11)$$

This configuration gives immediately

$$V^p_k = \frac{2k+1}{d} \qquad (4.2\text{-}12)$$

and, under the previous assumptions of Equation (4.2-1) about d, leads to the following approximations:

$$H^p_{k+1} = H^{p-1}_k + V^p_k - V^{p-1}_k \approx V^p_k = \frac{2k+1}{d} \qquad (4.2\text{-}13)$$

$$\rho^p_{2(k+1)} = \rho^{p-1}_{2k} + \frac{2(k+1)}{H^p_{k+1}} \approx \rho^{p-1}_{2k} + \left(\frac{2k+2}{2k+1}\right) d \qquad (4.2\text{-}14a)$$

$$= \rho^p_{2k} + \frac{d}{2k+1} \qquad (4.2\text{-}14b)$$

$$= \rho^p_{2k} + \frac{1}{V^p_k} \qquad (4.2\text{-}14c)$$

Along the next upsloping diagonal, indexed by p+1, the approximations become

$$H^{p+1}_{k+1} = H^p_k + V^{p+1}_k - V^p_k \approx - V^p_k = - \frac{2k+1}{d} \qquad (4.2\text{-}15)$$

$$\rho^{p+1}_{2(k+1)} = \rho^{p}_{2k} + \frac{2(k+1)}{H^{p+1}_{k+1}} \approx \rho^{p}_{2k} - \frac{2k+2}{2k+1}\, d \tag{4.2-16a}$$

$$= \rho^{p-1}_{2k} - \frac{d}{2k+1} \tag{4.2-16b}$$

$$= \rho^{p-1}_{2k} - \frac{1}{V^{p}_{k}} \tag{4.2-16c}$$

$$V^{p+1}_{k+1} = \frac{2k+3}{\rho^{p+1}_{2(k+1)} - \rho^{p}_{2(k+1)}} \approx \frac{2k+3}{\left(\rho^{p}_{2k} - \frac{2k+2}{2k+1}\, d\right) - \left(\rho^{p}_{2k} + \frac{d}{2k+1}\right)}$$

$$= -\frac{(2k+1)}{d} = -V^{p}_{k} \tag{4.2-17}$$

$$H^{p+1}_{k+2} = \underline{H^{p}_{k+i} + V^{p+1}_{k+1}} - V^{p}_{k+1} \approx -V^{p}_{k+1} \tag{4.2-18}$$

$$\rho^{p+1}_{2(k+2)} = \rho^{p}_{2(k+1)} + \frac{2(k+2)}{H^{p+1}_{k+2}} = \rho^{p}_{2(k+1)} - \frac{2(k+2)}{V^{p}_{k+1}} \tag{4.2-19a}$$

$$= \rho^{p-1}_{2(k+1)} + \frac{2k+3}{V^{p}_{k+1}} - \frac{2k+4}{V^{p}_{k+1}} = \rho^{p-1}_{2(k+1)} - \frac{1}{V^{p}_{k+1}} \tag{4.2-19b}$$

Still another upsloping diagonal is also affected, that indexed by p+2:

$$H^{p+2}_{k+2} = \underline{H^{p+1}_{k+1}} + V^{p+2}_{k+1} - \underline{V^{p+1}_{k+1}} \approx -\underline{V^{p}_{k}} + V^{p+2}_{k+1} + \underline{V^{p}_{k}} = V^{p+2}_{k+1} \tag{4.2-20}$$

$$\rho^{p+2}_{2(k+2)} = \rho^{p+1}_{2(k+1)} + \frac{2(k+2)}{H^{p+2}_{k+2}} \approx \rho^{p+1}_{2(k+1)} + \frac{2(k+2)}{V^{p+2}_{k+1}} \tag{4.2-21a}$$

$$= \rho^{p+2}_{2(k+1)} - \frac{2k+3}{V^{p+2}_{k+1}} + \frac{2k+4}{V^{p+2}_{k+1}} = \rho^{p+2}_{2(k+1)} + \frac{1}{V^{p+2}_{k+1}} \tag{4.2-21b}$$

Again, the corresponding singular rules are considerably simpler: for d = 0,

$$\rho^{p-1}_{2k} = \rho^{p}_{2k} = \rho^{p}_{2(k+1)} = \rho^{p+1}_{2(k+1)} = \rho\text{, say;} \tag{4.2-22}$$

$$V^{p}_{k} = V^{p+1}_{k+1} = H^{p}_{k+1} = H^{p+1}_{k+1} = \infty \tag{4.2-23}$$

$$\rho^{p+1}_{2(k+2)} = \rho - \frac{2(k+2)}{V^{p}_{k+1}} \tag{4.2-24}$$

$$\rho^{p+2}_{2(k+2)} = \rho + \frac{2(k+2)}{V^{p+2}_{k+1}} \tag{4.2-25}$$

In Figure 4.2, the very heavy dots indicate those tabular entries of rho or epsilon which are involved in the second case of the near-singular rules. Heavy bars between the heavy dots indicate very large values of the corresponding H or V; these values become infinite in the singular case.

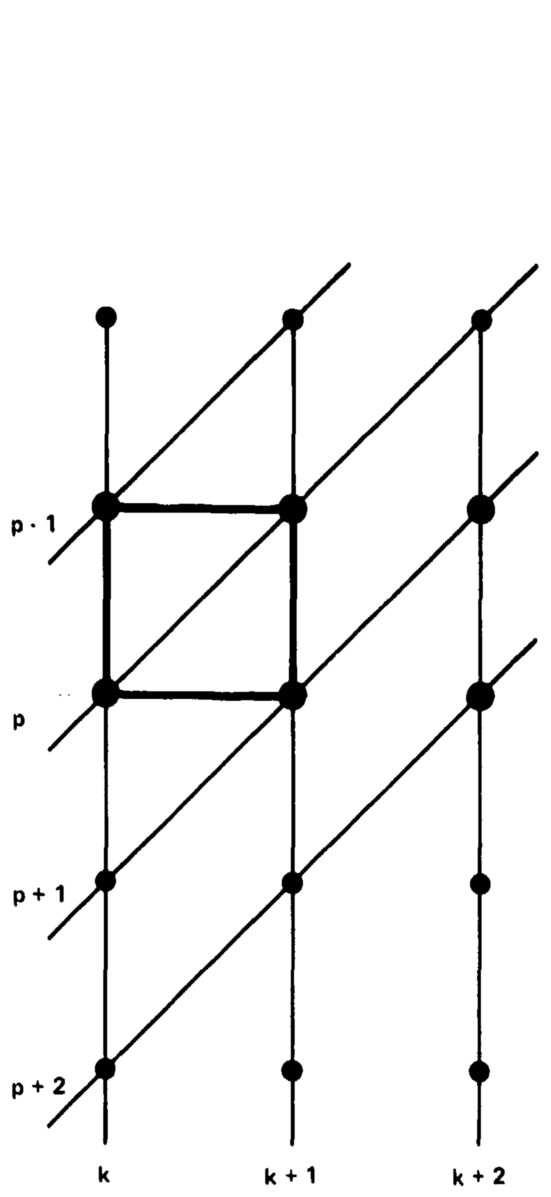


Figure 4.2 - Second Case of Near-Singular Rules for the Even-Rho and Even-Epsilon Algorithms

4.3 NEAR-SINGULAR RULES FOR THE EVEN-EPSILON ALGORITHM

The opening remarks in Section 4.2, and in particular the restrictions of Equation (4.2-1) on the small divisor, d, apply as well to the even-epsilon algorithm. The only difference is that now H and V are defined by Equations (3.3-1) and (3.3-2), respectively, and the whole algorithm is given by Equations (3.3-3) thru (3.3-7). To facilitate comparisons between the corresponding near-singular rules for the even-rho and even-epsilon algorithms, corresponding formulas bear corresponding numbers. Equation (4.3-1) is missing but it would be identical to Equation (4.2-1).

In the first case of near singularity, let

$$H^{p}_{k+1} = H^{p-1}_{k} + V^{p}_{k} - V^{p-1}_{k} = d \tag{4.3-2}$$

where d meets the restrictions of Equation (4.2-1). Then Equation (3.3-1), the definition of H for the even-epsilon algorithm, gives

$$\varepsilon^{p}_{2(k+1)} = \varepsilon^{p-1}_{2k} + \frac{1}{H^{p}_{k+1}} \approx \frac{1}{d} \tag{4.3-3}$$

and Equation (3.3-2), the definition of V, gives

$$V^{p}_{k+1} = \frac{1}{\varepsilon^{p}_{2(k+1)} - \varepsilon^{p-1}_{2(k+1)}} \approx d \tag{4.3-4}$$

On the next upsloping diagonal, indexed by p+1,

$$V^{p+1}_{k+1} = \frac{1}{\varepsilon^{p+1}_{2(k+1)} - \varepsilon^{p}_{2(k+1)}} \approx - d \tag{4.3-5}$$

$$H^{p+1}_{k+2} = H^{p}_{k+1} + V^{p+1}_{k+1} - V^{p}_{k+1} \approx d - 2d = - d \tag{4.3-6}$$

$$\varepsilon^{p+1}_{2(k+2)} = \varepsilon^{p}_{2(k+1)} + \frac{1}{H^{p+1}_{k+2}} \approx \varepsilon^{p-1}_{2k} + \underline{\frac{1}{d} - \frac{1}{d}} = \varepsilon^{p-1}_{2k} \tag{4.3-7}$$

These results are summarized in Figure 4.3:

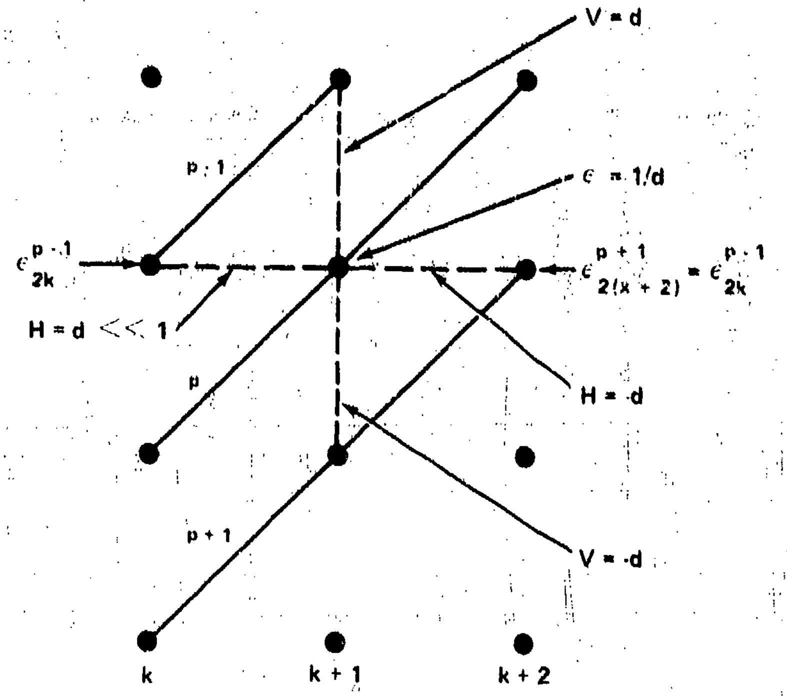


Figure 4.3 - First Case of Near-Singular Rules for the Even-Epsilon Algorithm

The corresponding singular rules are, for d=0,

$$V_{k+1}^{p} = V_{k+1}^{p+1} = H_{k+1}^{p} = H_{k+2}^{p+1} = 0 \tag{4.3-8}$$

$$\varepsilon_{2(k+1)}^{p} = \infty \tag{4.3-9}$$

$$\varepsilon_{2(k+2)}^{p+1} = \varepsilon_{2k}^{p-1} \tag{4.3-10}$$

The second and more complicated case of near singularity arises when a newly calculated ε_{2k}^{p} is almost equal to ε_{2k}^{p-1} directly above it in the even-epsilon array:

$$\varepsilon_{2k}^{p} - \varepsilon_{2k}^{p-1} = d \tag{4.3-11}$$

This configuration gives immediately

$$V_k^p = \frac{1}{d} \qquad (4.3\text{-}12)$$

and, under the stated assumptions of Equation (4.2-1) about d, leads to the following approximations:

$$H_{k+1}^p = H_k^{p-1} + V_k^p - V_k^{p-1} \approx V_k^p = \frac{1}{d} \qquad (4.3\text{-}13)$$

$$\varepsilon_{2(k+1)}^p = \varepsilon_{2k}^{p-1} + \frac{1}{H_{k+1}^p} \approx \varepsilon_{2k}^{p-1} + d = \varepsilon_{2k}^p \qquad (4.3\text{-}14)$$

Along the next upsloping diagonal, indexed by p+1, the approximations become

$$H_{k+1}^{p+1} = H_k^p + V_k^{p+1} - V_k^p \approx - V_k^p = - \frac{1}{d} \qquad (4.3\text{-}15)$$

$$\varepsilon_{2(k+1)}^{p+1} = \varepsilon_{2k}^p + \frac{1}{H_{k+1}^{p+1}} \approx \varepsilon_{2k}^p - d = \varepsilon_{2k}^{p-1} \qquad (4.3\text{-}16)$$

$$V_{k+1}^{p+1} = \frac{1}{\varepsilon_{2(k+1)}^{p+1} - \varepsilon_{2(k+1)}^p} \approx \frac{1}{\varepsilon_{2k}^p - d - \varepsilon_{2k}^p} = - \frac{1}{d} = - V_k^p \qquad (4.3\text{-}17)$$

$$H_{k+2}^{p+1} = \underline{H_{k+1}^p + V_{k+1}^{p+1}} - V_{k+1}^p \approx - V_{k+1}^p \qquad (4.3\text{-}18)$$

$$\varepsilon_{2(k+2)}^{p+1} = \varepsilon_{2(k+1)}^p + \frac{1}{H_{k+2}^{p+1}} = \varepsilon_{2(k+1)}^p - \frac{1}{V_{k+1}^p} \qquad (4.3\text{-}19)$$

Still another upsloping diagonal is also affected, that indexed by p+2:

$$H^{p+2}_{k+2} = \underline{H^{p+1}_{k+1}} + V^{p+2}_{k+1} - \underline{V^{p+1}_{k+1}} \approx -\underline{\frac{1}{d}} + V^{p+2}_{k+1} + \underline{\frac{1}{d}} = V^{p+2}_{k+1} \qquad (4.3\text{-}20)$$

$$\varepsilon^{p+2}_{2(k+2)} = \varepsilon^{p+1}_{2(k+1)} + \frac{1}{H^{p+2}_{k+2}} \approx \varepsilon^{p+1}_{2(k+1)} + \frac{1}{V^{p+2}_{k+1}} \qquad (4.3\text{-}21)$$

The corresponding singular rules, for $d = 0$, are

$$\varepsilon^{p-1}_{2k} = \varepsilon^{p}_{2k} = \varepsilon^{p}_{2(k+1)} = \varepsilon^{p+1}_{2(k+1)} = \varepsilon, \text{ say;} \qquad (4.3\text{-}22)$$

$$V^{p}_{k} = V^{p+1}_{k+1} = H^{p}_{k+1} = H^{p+1}_{k+1} = \infty \qquad (4.3\text{-}23)$$

$$\varepsilon^{p+1}_{2(k+2)} = \varepsilon - \frac{1}{V^{p}_{k+1}} \qquad (4.3\text{-}24)$$

$$\varepsilon^{p+2}_{2(k+2)} = \varepsilon + \frac{1}{V^{p+2}_{k+1}} \qquad (4.3\text{-}25)$$

5. IMPLEMENTATION OF THE ALGORITHMS

5.1 COMPUTER PROGRAMS

In order to carry out computational experiments, a small package of computer programs was developed. The even-rho algorithm, as described in Section 2.4, was implemented as subroutine EVRHO in CDC FORTRAN Extended (essentially FORTRAN IV). Similarly, the even-epsilon algorithm described in Section 3.3 was implemented as subroutine EVEPS. In addition, a merged version of these subroutines, called TANDEM, was written to take advantage of the fact that they were almost identical, differing only in two lines of coding. The price was to rename most of the intermediate quantities and to double the amount of storage required which was trivial. These subroutines did not incorporate the singular rules or near-singular rules described in Section 4 because they were written before the need for near-singular rules was perceived and because the necessary modifications would have been relatively complicated. Protection was built in, however, which aborted computations whenever a divisor became too small.

Each sequence to be extrapolated requires the writing of a brief subroutine SEQU to calculate the successive numbers of the sequence. Five examples are listed in Appendix A. In each case the sequence is identified in the included comment cards.

In addition, there was an executive routine TDMCHK to perform the usual chores of calling subroutines and handling output.

Listings of all of these programs are given in Appendix A.

5.2 NUMERICAL EXAMPLES

Corresponding to each of the sequence generating subroutines SEQU listed in Appendix A, is an output page in Appendix B which shows how EVRHO and EVEPS performed in that sequence. As might be expected from the theoretical discussions in Sections 2.1 and 3.1, these two algorithms performed rather differently, one being superior to the other for any particular sequence but neither being consistently superior for all sequences. In the fourth example neither performed very well.

ACKNOWLEDGMENTS

To Daniel Shanks I owe a considerable debt of gratitude for friendly encouragement and intellectual stimulation. It was my good fortune to be a friend and colleague of his during the early 1950's when his transforms were new and exciting. From him, too, I learned about the usefulness of continued fractions, rational approximations, the Padé Table, and related matters. In 1975, when my interest in these subjects was rekindled, he gave me free access to his magnificent library and especially to the publications of Peter Wynn and Claude Brezinski.

Correspondence with Claude Brezinski has ripened into friendship. To him, I am indebted for much needed encouragement and stimulation as well as for copies of many publications by himself and his students.

APPENDIX A

LISTINGS OF COMPUTER PROGRAMS

IDMCHK:	Control routine
EVEPS:	Subroutine for even-epsilon algorithm
EVRHO:	Subroutine for even-rho algorithm
TANDEM:	Merged version of EVEPS and EVRHO
SEQU:	Sequence generators for:

(1) $s_n = (1+1/n)^n$

(2) $s_n = \sum_{k=1}^{n} (-1)^{k+1}/k$

(3) $s_n = \sum_{k=1}^{n} 1/k^2$

(4) s_n = First difference of logarithm of Turning condition number for leading $n \times n$ segment of the Hilbert matrix. Theory from Todd (1954),[24] data from Savage and Lukacs (1945).[25]

(5) s_n = First difference of logarithm of Turning condition number for leading $n \times n$ segment of the Hilbert matrix. Theory from Todd (1954),[24] data from Fettis and Caslin (1967).[26]

```
           PROGRAM TOMCHK(INPUT, OUTPUT, TAPE5 = INPUT, TAPE 6 = OUTPUT)
      C
      C    EXERCISE SUBROUTINE TANDEM FOR CALCULATING BOTH EVEN ORDERED
      C    EPSILONS AND EVEN ORDERED RECIPROCAL DIFFERENCES FOR VARIOUS
      C    INPUT SEQUENCES, SN.  THE LATTER ARE DEFINED BY A SEPARATE
      C    SUBROUTINE, SEQU.
      C
      C    THE QUANTITIES ENEW, EOLD, AND RNEW, ROLD ARE FOR USE IN TESTING
      C    FOR CONVERGENCE, IF SO DESIRED.
      C
      C    NS IS THE NUMBER OF SKIPS BEFORE CALLING TANDEM.
      C    NP IS THE MAXIMUM VALUE OF THE ITERATION INDEX, NZ.
      C    NP - NS = MAXIMUM VALUE OF INDEX, IP.
      C
      C    IP MUST BE SET TO ZERO ON INITIAL CALL TO ASSURE A PROPER START.
           IP = 0
           NS = 0
           NP = 40
           SNEW = 0.0
           ENEW = 0.0
           RNEW = 0.0
        10 WRITE(6,1010)
      1010 FORMAT(49H1EVEN ORDERED EPSILONS AND RECIPROCAL DIFFERENCES)
        11 WRITE(6,1011)
      1011 FORMAT(59H SEQUENCE SN = SUM(1/N**2).....((PI)**2)/6 = 1.644934066
          1848)
        12 WRITE(6,1012)
      1012 FORMAT(39H0EVEN ORDER = 2K FOR IP = 2K+1 AND 2K+2//3H IP,20X,5HINP
          1UT,18X,7HEPSILON,22X,3HRHO/)
        15 DO 50 NZ = 1,NP
           EOLD = ENEW
           ROLD = RNEW
           SOLD = SNEW
        20 CALL SEQU(NZ,SOLD,SNEW)
           IF(NZ .LE. NS) GO TO 50
           SN = SNEW
        25 CALL EVEPS(SN,EP,IP,NS,IR)
           CALL EVRHO(SN,RH,IP,NS,IR)
           ENEW = EP
           RNEW = RH
           IF(IR .EQ. 1) GO TO 100
        30 WRITE(6,1030) IP,SN,EP,RH
      1030 FORMAT(1H ,I2,3E25.14)
        50 CONTINUE
       100 WRITE(6,1100)
      1100 FORMAT(9H0FINISHED)
           STOP
           END
      SUBROUTINE EVEPS(SN,EP,IP,NS,IR)
C
C     EVEPS COMPUTES AN ARRAY OF EVEN ORDERED EPSILONS, R1(.), FOR A
C     GIVEN SEQUENCE, SN, USING THE RECURSION RELATION GIVEN BY PETER WYNN
C     FOR THE EPSILON ARRAY AND THE PADE TABLE.
C     (NUM MATH 8 (1966) 264-269)
C
      DIMENSION H1(50),H2(50),R1(50),R2(50),V1(50),V2(50)
C     IF 2*KMAX IS THE HIGHEST ORDERED RECIPROCAL DIFFERENCE DESIRED,
C     THEN THE DIMENSION OF THESE ARRAYS MUST BE AT LEAST KMAX + 1.
C
C     THE HIGHEST ORDER OF EPSILON CALCULATED AND SENT TO OUTPUT IS
C     2K FOR IP = 2K+1 AND 2K+2.
C
C     IP MUST BE SET TO ZERO ON INITIAL CALL TO ASSURE A PROPER START.
C
      IR = 0
C     IR IS AN ERROR INDICATOR WHICH IS SET TO 1 IN CASE OF PENDING
C     DIVISION BY ZERO.  IT MUST BE TESTED AND PROPER ACTION TAKEN IN
C     THE CALLING PROGRAM.
C
      TOL = 1.0E-14
   10 IP = IP + 1
      IF(IP-2) 12,11,20
   11 V2(1) = 1.0/(SN - R2(1))
   12 R2(1) = SN
      H2(1) = 0.0
      GO TO 100
   20 JMAX = (IP+1)/2
      R1(1) = SN
      H1(1) = 0.0
      DO 25 J=1,JMAX
      WS = R1(J) - R2(J)
      IF(ABS(WS) .LT. TOL) GO TO 95
   21 V1(J) = 1.0/WS
      WS = H2(J)
      IF(J .EQ. 1) WS = 0.0
   23 H1(J+1) = WS + V1(J) - V2(J)
      IF(ABS(H1(J+1)) .LT. TOL) GO TO 95
   25 R1(J+1) = R2(J) + 1.0/H1(J+1)
   30 EP = R1(JMAX)
   40 DO 43 J=1,JMAX
      H2(J) = H1(J)
      R2(J) = R1(J)
   43 V2(J) = V1(J)
      GO TO 100
   95 WRITE(6,1095) IP,J
 1095 FORMAT(26H0DIVISOR TOO SMALL AT IP =,I3,6H,   J =,I2)
      IR = 1
  100 RETURN
      END
```

```
SUBROUTINE EVRHO      73/74   OPT=0  ROUND=+-*/  TRACE          FTN 4.6+420

      SUBROUTINE EVRHO(SN,RH,IP,WS,IR)
C
C     EVRHO COMPUTES AN ARRAY OR EVEN ORDERED RECIPROCAL DIFFERENCES,
C     R1(.), FOR A GIVEN SEQUENCE, SN, USING A RECURSION RELATION ANALOGOUS
C     TO THAT GIVEN BY PETER WYNN FOR THE EPSILON ARRAY AND THE PADE TABLE.
C     (NUM MATH 8 (1966) 264-269)
C
      DIMENSION H1(50),H2(50),R1(50),R2(50),V1(50),V2(50)
C     IF 2*KMAX IS THE HIGHEST ORDERED RECIPROCAL DIFFERENCE DESIRED,
C     THEN THE DIMENSION OF THESE ARRAYS MUST BE AT LEAST KMAX + 1.
C
C     THE HIGHEST ORDER RECIPROCAL DIFFERENCE CALCULATED AND SENT TO OUTPUT IS
C     2K FOR IP = 2K+1 AND 2K+2.
C
C     IP MUST BE SET TO ZERO ON INITIAL CALL TO ASSURE A PROPER START.
C
      IR = 0
C     IR IS AN ERROR INDICATOR WHICH IS SET TO 1 IN CASE OF FENDING
C     DIVISION BY ZERO.  IT MUST BE TESTED AND PROPER ACTION TAKEN IN
C     THE CALLING PROGRAM.
C
      TOL = 1.0E-14
      IF(IP-2) 12,11,20
   11 V2(1) = 1.0/(SN - R2(1))
   12 R2(1) = SN
      H2(1) = 0.0
      GO TO 100
   20 JMAX = (IP+1)/2
      R1(1) = SN
      H1(1) = 0.0
      DO 25 J=1,JMAX
      WS = R1(J) - R2(J)
      IF(ABS(WS) .LT. TOL) GO TO 95
   21 V1(J) = (2*J-1)/WS
      WS = H2(J)
      IF(J .EQ. 1) WS = 0.0
   23 H1(J+1) = WS + V1(J) - V2(J)
      IF(ABS(H1(J+1)) .LT. TOL) GO TO 95
   25 R1(J+1) = R2(J) + (2*J)/H1(J+1)
   30 RH = R1(JMAX)
   40 DO 43 J=1,JMAX
      H2(J) = H1(J)
      R2(J) = R1(J)
   43 V2(J) = V1(J)
      GO TO 100
   95 WRITE(6,1095) IP,J
 1095 FORMAT(26H0DIVISOR TOO SMALL AT IP =,I3,6H,  J =,I2)
      IR = 1
  100 RETURN
      END
```

```
                 SUBROUTINE TANDEM(SN,EP,RH,IP,IR)
          C
          C      TANDEM COMPUTES TWO ARRAYS SIMULTANEOUSLY FROM A GIVEN INPUT SEQUENCE,
          C      SN.  EVEN ORDERED EPSILONS IN RE1 ARE COMPUTED USING A RECURSION
          C      RELATION GIVEN BY PETER WYNN FOR THE EPSILON ARRAY AND PADE TABLE.
          C      (NUM MATH 8 (1966) 264-269)
          C      EVEN ORDERED RECIPROCAL DIFFERENCES IN RR1 ARE COMPUTED USING AN
          C      ANALOGOUS RECURSION RELATION DEVELOPED BY R. P. EDDY. (UNPUBLISHED)
          C
                 DIMENSION HE1(50), HE2(50), RE1(50), RE2(50), VE1(50), VE2(50)
                 DIMENSION HR1(50), HR2(50), RR1(50), RR2(50), VR1(50), VR2(50)
          C
          C      RELATIONS AMONG PARAMETERS.....
          C      LET ND BE THE DIMENSION OF THESE ARRAYS, AND LET 2*KMAX BE THE
          C      HIGHEST ORDERED EPSILON AND/OR RECIPROCAL DIFFERENCE DESIRED.
          C      THEN ND .GE. KMAX+1
          C      FURTHERMORE, MAX IP = NP-NS = 2*KMAX+2 .LE. 2*ND
          C      EXAMPLE...  ND = 50,  2*KMAX = 98,  MAX IP = 100,  NP = 100,  NS = 0
          C
          C      CONVERSELY, FOR A GIVEN IP, THE HIGHEST ORDER OF EPSILON AND/OR
          C      RECIPROCAL DIFFERENCE CALCULATED AND SENT TO OUTPUT IS
          C      2K FOR IP = 2K+1 AND 2K+2.
          C
          C      IP MUST BE SET TO ZERO ON INITIAL CALL TO ASSURE A PROPER START.
          C
                 IR = 0
          C      IR IS AN ERROR INDICATOR WHICH IS SET TO 1 IN CASE OF PENDING
          C      DIVISION BY ZERO.  IT MUST BE TESTED AND PROPER ACTION TAKEN IN
          C      THE CALLING PROGRAM.
          C
                 TOL = 1.0E-14
              10 IP = IP + 1
                 IF(IP-2) 12,11,20
              11 VE2(1) = 1.0/(SN - RE2(1))
                 VR2(1) = VE2(1)
              12 RE2(1) = SN
                 RR2(1) = SN
                 HE2(1) = 0.0
                 HR2(1) = 0.0
                 GO TO 100
              20 JMAX = (IP+1)/2
                 RE1(1) = SN
                 RR1(1) = SN
                 HE1(1) = 0.0
                 HR1(1) = 0.0
                 DO 25 J=1,JMAX
                 WE = RE1(J) - RE2(J)
                 WR = RR1(J) - RR2(J)
                 IF(ABS(WE) .LT. TOL .OR. ABS(WR) .LT. TOL) GO TO 95
              21 VE1(J) = 1.0/WE
                 VR1(J) = (2*J-1)/WR
                 WE = HE2(J)
                 WR = HR2(J)
                 IF(J .NE. 1) GO TO 23
                 WE = 0.0
                 WR = 0.0
              23 HE1(J+1) = WE + VE1(J) - VE2(J)
```

```
               HR1(J+1) = HR + VR1(J) - VR2(J)
               IF(ABS(HE1(J+1)) .LT. TOL .OR. ABS(HR1(J+1)) .LT. TOL) GO TO 95
               RE1(J+1) = RE2(J) + 1.0/HE1(J+1)
            25 RR1(J+1) = RR2(J) + (2*J)/HR1(J+1)
            30 EP = RE1(JMAX)
               RH = RR1(JMAX)
            40 DO 43 J=1,JMAX
               HE2(J) = HE1(J)
               HR2(J) = HR1(J)
               RE2(J) = RE1(J)
               RR2(J) = RR1(J)
               VE2(J) = VE1(J)
            43 VR2(J) = VR1(J)
               GO TO 100
            95 WRITE(6,1095) IP,J
          1095 FORMAT(26H0DIVISOR TOO SMALL AT IP =,I3,6H,  J =,I2)
               IR = 1
           100 RETURN
               END
```

```
SUBROUTINE SEQU       73/74   OPT=0 ROUND=+-*/ TRACE          FTN 4.6+420

                  SUBROUTINE SEQU(NZ,SOLD,SNEW)
C                 CALCULATION OF SEQUENCE (1+1/N)**N, THE LIMIT OF WHICH IS
C                 E = 2.718281828459045.....
C
                  PRD = 1.0
                  FAC = (1.0 + 1.0/NZ)
                  DO 10 N=1,NZ
               10 PRD = PRD*FAC
                  SNEW = PRD
              100 RETURN
                  END
```

```
SUBROUTINE SEQU        73/74    OPT=0 ROUND=+-*/ TRACE          FTN 4.6+420        01/14/77

                      SUBROUTINE SEQU(NZ,SOLD,SNEW,SW)
                C     PARTIAL SUMS OF SUM(((-1)**(N-1))/N).....LN(2) = 0.693147180559945.....
                C     SW IS SWITCH TO EFFECT ALTERNATING SIGN IN SUMMATION.
                   10 SW = -SW
                   11 SNEW = SOLD + SW/NZ
                  100 RETURN
                      END
```

```
SUBROUTINE SEQU        73/74    OPT=0 ROUND=+-*/ TRACE          FTN 4.6+420

          SUBROUTINE SEQU(NZ,SOLD,SNEW)
C         PARTIAL SUMS SN = SUM(1/N**2)  FROM 1 TO NZ.
C         LIMIT IS (PI**2)/6 = 1.64493 40668 48...
C
     10   SNEW = SOLD + 1.0/NZ**2
    100   RETURN
          END
```

```
SUBROUTINE SEQU       73/74   OPT=0 ROUND=+-*/ TRACE          FTN 4.6+420

                 SUBROUTINE SEQU(NZ,SOLD,SNEW)
           C     ASYMPTOTIC TURING CONDITION NUMBER OF LEADING SEGMENTS OF THE
           C     HILBERT MATRIX.....EXP(3.525*N)
                 DIMENSION ELMX(10)
                 ELMX( 1) = 1.
                 ELMX( 2) = 12.
                 ELMX( 3) = 192.
                 ELMX( 4) = 6480.
                 ELMX( 5) = 179200.
                 ELMX( 6) = 4410000.
                 ELMX( 7) = 133402500.
                 ELMX( 8) = 4249941696.
                 ELMX( 9) = 122367445200.
                 ELMX(10) = 3480673996800.
              10 SNEW = ALOG((NZ*ELMX(NZ))/((NZ-1)*ELMX(NZ-1)))
             100 RETURN
                 END
```

```
SUBROUTINE SEQU        73/74    OPT=0 ROUND=+-*/ TRACE            FTN 4.6+420

                 SUBROUTINE SEQU(NZ,SOLD,SNEW)
          C      ASYMPTOTIC SPECTRAL CONDITION NUMBER OF LEADING SEGMENTS OF THE
          C      HILBERT MATRIX.....EXP(3.525*N)
                 DIMENSION EVL1(10), EVLN(10)
                  VL1( 1) = 1.0
                 EVL1( 2) = 2.0/3.0 + (SQRT(13.0))/6.0
                 EVL1( 3) = 1.4083189271237  E0
                 EVL1( 4) = 1.5002142800592  E0
                 EVL1( 5) = 1.5670506910982  E0
                 EVL1( 6) = 1.6188998589243  E0
                 EVL1( 7) = 1.6608853389269  E0
                 EVL1( 8) = 1.6959389969219  E0
                 EVL1( 9) = 1.7258826639018  E0
                 EVL1(10) = 1.7519196732652  E0
                 EVLN( 1) = 1.0
                 EVLN( 2) = 2.0/3.0 - (SQRT(13.0))/6.0
                 EVLN( 3) = 2.6873403557735  E-03
                 EVLN( 4) = 9.6702304022587  E-05
                 EVLN( 5) = 3.2879287721719  E-06
                 EVLN( 6) = 1.0827994845656  E-07
                 EVLN( 7) = 3.4938986059912  E-09
                 EVLN( 8) = 1.1115389663724  E-10
                 EVLN( 9) = 3.4936764029115  E-12
                 EVLN(10) = 1.0931538193797  E-13
              10 A = (EVL1(NZ)*EVLN(NZ-1))/(EVLN(NZ)*EVL1(NZ-1))
              11 SNEW = ALOG(A)
             100 RETURN
                 END
```

APPENDIX B
SAMPLE OUTPUT

Output pages show the performance of EVEPS and EVRHO on each of the sequences given in Appendix A. The first three sequences are well known, as are their limits:

(1) $e = 2.71828\ 18284\ 59045$

(2) $\ln 2 = 0.69314\ 71805\ 59945$

(3) $\pi^2/6 = 1.64493\ 40668\ 48226$

(4) Following Todd[24] one can show that, for large n, the Turning condition number of leading segments of the Hilbert matrix is asymptotically

$$C_T \sim (1/4\pi^2\sqrt{2})\ \mathrm{EXP}(nB)$$

where

$$B = 2\ln\ ((\sqrt{2}+1)/(\sqrt{2}-1)) = 3.525496$$

This B is the limit of the sequence s_n.

(5) It is conjecturable that, asymptotically, the spectral condition number is the same as the Turning condition number for leading segments of the Hilbert matrix. This seems to be borne out by numerical evidence from the even-rho algorithm.

```
EVEN ORDERED EPSILONS AND RECIPROCAL DIFFERENCES
SEQUENCE SN =  (1+1/N)**N    ...E = 2.718281828459045

EVEN ORDER 2K = 2*(INT((IP-~~N~~S+1)/2)-1)

IP                INPUT                    EPSILON                      RHO

 1      .20000000000000E+01    0.                          0.
 2      .22500000000000E+01    0.                          0.
 3      .23703703703704E+01     .24821428571429E+01        .27142857142858E+01
 4      .24414062500000E+01     .25436895840121E+01        .27170087976538E+01
 5      .24883200000000E+01     .26011604792044E+01        .27182741116763E+01
 6      .25216263717421E+01     .26255330464090E+01        .27182801209970E+01
 7      .25464996970408E+01     .26483852074337E+01        .27182818200403E+01
 8      .25657845139503E+01     .26604813430328E+01        .27182818268947E+01
 9      .25811747917133E+01     .26718636223536E+01        .27182818291366E+01
10      .25937424601000E+01     .26787352804142E+01        .27182818277979E+01
11      .26041990118976E+01     .26852227048524E+01        .27182818285893E+01
12      .26130352902248E+01     .26894967513636E+01        .27182818282838E+01
13      .26206008878858E+01     .26935445952391E+01        .27182818284826E+01
14      .26271515563008E+01     .26963809489453E+01        .27182818283089E+01
15      .26328787177280E+01     .26990837655430E+01        .27182818284776E+01
16      .26379284973666E+01     .27010217152073E+01        .27182818283831E+01
17      .26424143751832E+01     .27032533190518E+01        .27182818284726E+01
18      .26464258210978E+01     .27033996217623E+01        .27182818284336E+01
19      .26500343266405E+01     .27032400441073E+01        .27182818284561E+01
20      .26532977051443E+01     .27053337423285E+01        .27182818284320E+01

FINISHED
```

```
EVEN ORDERED EPSILONS AND RECIPROCAL DIFFERENCES
SEQUENCE SN = SUM(((-1)**(N-1))/N).....LN(2) = 0.6931471805599945...

EVEN ORDER = 2K FOR IP = 2K+1 AND 2K+2

IP              INPUT                     EPSILON                        RHO

 1       .10000000000000E+01      0.                            0.
 2       .50000000000000E+00      0.                            0.
 3       .83333333333334E+00       .70000000000000E+00           .90000000000000E+00
 4       .58333333333334E+00       .69047619047619E+00           .54761904761905E+00
 5       .78333333333333E+00       .69333333333334E+00           .85193386477709E+00
 6       .61666666666667E+00       .69308943089431E+00           .57479173257408E+00
 7       .75952380952381E+00       .69315245478036E+00           .82309710153248E+00
 8       .63452380952381E+00       .69314574314575E+00           .59268011878977E+00
 9       .74563492063492E+00       .69314733235438E+00           .80366273568942E+00
10       .64563492063492E+00       .69314714248772E+00           .60547922746173E+00
11       .73654401154401E+00       .69314718496213E+00           .78958444622312E+00
12       .65321067821068E+00       .69314717951778E+00           .61515318836070E+00
13       .73013375513375E+00       .69314718068816E+00           .77886993154328E+00
14       .65870518370518E+00       .69314718053085E+00           .62275567853094E+00
15       .72537185037185E+00       .69314718056369E+00           .77041671569557E+00
16       .66287185037185E+00       .69314718055912E+00           .62890724450765E+00
17       .72169537978361E+00       .69314718056035E+00           .76356207517815E+00
18       .66613982422805E+00       .69314718055992E+00           .63399917173487E+00
19       .71877140317542E+00       .69314718055934E+00           .75788217418649E+00

DIVISOR TOO SMALL AT IP = 20,  J = 9

FINISHED
```

```
EVEN ORDERED EPSILONS AND RECIPROCAL DIFFERENCES
SEQUENCE SN = SUM(1/N**2)......((PI)**2)/6 = 1.6449340668482226

EVEN ORDER = 2K FOR IP = 2K+1 AND 2K+2

IP                 INPUT                        EPSILON                          RHO

 1     .10000000000000E+01      0.                               0.
 2     .12500000000000E+01      0.                               0.
 3     .13611111111111E+01       .14500000000000E+01              .16500000000000E+01
 4     .14236111111111E+01       .15039682539683E+01              .16468253968254E+01
 5     .14636111111111E+01       .15516174402250E+01              .16448948948949E+01
 6     .14913888888889E+01       .15717673885475E+01              .16449225865211E+01
 7     .15117970521542E+01       .15903054136156E+01              .16449343761237E+01
 8     .15274220521542E+01       .15999841551501E+01              .16449341453734E+01
 9     .15397677311665E+01       .16090869062916E+01              .16449340643967E+01
10     .15497677311665E+01       .16144742951860E+01              .16449340662193E+01
11     .15580321939765E+01       .16196099139024E+01              .16449340673419E+01
12     .15649766384209E+01       .16229152906156E+01              .16449340665627E+01
13     .15708937981842E+01       .16260947416967E+01              .16449340669135E+01
14     .15759958390005E+01       .16282682200907E+01              .16449340667654E+01
15     .15804402834450E+01       .16303725523874E+01              .16449340669994E+01
16     .15843465334450E+01       .16318771932900E+01              .16449340668089E+01
17     .15878067410574E+01       .16333452531306E+01              .16449340668418E+01
18     .15908931608105E+01       .16344173994823E+01              .16449340668014E+01
19     .15936632439130E+01       .16355339285028E+01              .16449340668470E+01
20     .15961632439130E+01       .16362259559140E+01              .16449340668398E+01
21     .15984308176092E+01       .16368694332669E+01              .16449340668470E+01
22     .16004969333116E+01       .16357597244504E+01              .16449340668428E+01
23     .16023872924799E+01       .16379715468924E+01              .16449340668466E+01
24     .16041234035910E+01       .16374404478456E+01              .16449340669004E+01
25     .16057234035910E+01       .16377011907147E+01              .16449340668476E+01
26     .16072026935318E+01       .16385374869639E+01              .16449340668185E+01
27     .16085744356443E+01       .16386790625905E+01              .16449340668463E+01
28     .16098499458484E+01       .16385973425784E+01              .16449340668457E+01
29     .16110390064905E+01       .16386565826394E+01              .16449340668464E+01
30     .16121501176016E+01       .16385434600996E+01              .16449340668429E+01
31     .16131907003279E+01       .16395586315824E+01              .16449340668462E+01
32     .16141672628279E+01       .16393734364764E+01              .16449340668462E+01
33     .16150855364735E+01       .16394875100907E+01              .16449340668462E+01
34     .16159505883766E+01       .16406008310537E+01              .16449340668519E+01
35     .16167669149072E+01       .16410161862069E+01              .16449340668462E+01
36     .16175385198455E+01       .16406407078414E+01              .16449340668465E+01
37     .16182689800354E+01       .16401147089310E+01              .16449340668492E+01
38     .16189615008110E+01       .16402954578813E+01              .16449340668464E+01
39     .16196189630069E+01       .16404572488894E+01              .16449340668463E+01
40     .16202439630069E+01       .16397048229620E+01              .16449340668465E+01

FINISHED
```

```
EVEN ORDERED EPSILONS AND RECIPROCAL DIFFERENCES
ASYMPTOTIC TURING CONDITION NUMBER OF LEADING SEGMENTS OF THE HILBERT MATRIX.....EXP(3.525*N)
                                                                                     3.52549?

EVEN ORDER = 2K FOR IP = 2K+1 AND 2K+2

IP                 INPUT                    EPSILON                        RHO

 1      .31780538303479E+01      0.                           0.
 2      .38066624897703E+01      0.                           0.
 3      .35429255414909E+01       .36208742390442E+01         .34350859883180E+01
 4      .33854490246940E+01       .31520709648043E+01         .27612163881177E+01
 5      .35636468643041E+01       .35184519955906E+01         .34649450098916E+01
 6      .35948210619250E+01       .35301691087997E+01         .36460481625740E+01
 7      .34779061355159E+01       .35264643239269E+01         .34576039796504E+01
 8      .34533133828012E+01       .35334914480444E+01         .33667075123839E+01

FINISHED
```

```
EVEN ORDERED EPSILONS AND RECIPROCAL DIFFERENCES
ASYMPTOTIC SPECTRAL CONDITION NUMBER OF LEADING SEGMENTS OF THE HILBERT MATRIX.....EXP(3.525*N)
                                                                                   3.525494

EVEN ORDER = 2K FOR IP = 2K+1 AND 2K+2

IP            INPUT                 EPSILON                    RHO

 1     .29591445346109E+01     0.                        0.
 2     .33024554980121E+01     0.                        0.
 3     .33878812479438E+01     .34161789492862E+01       .35299024005603E+01
 4     .34249667753909E+01     .34534179548712E+01       .35189546617985E+01
 5     .34458446113850E+01     .34783158329939E+01       .35207577357080E+01
 6     .34593057746753E+01     .34893803854012E+01       .35257159029628E+01
 7     .34687435445766E+01     .34992727975061E+01       .35257357425462E+01
 8     .34757472285192E+01     .35036994108080E+01       .35257131922343E+01
 9     .34811622043110E+01     .35090779984178E+01       .35251103382613E+01

FINISHED
```

REFERENCES

1. Wynn, P., "Upon Systems of Recursions which Obtain among the Quotients of the Padé Table," Numerische Mathematik, Vol. 8, No. 3, pp. 264-269 (1966).

2. Thiele, T. N., "Interpolationsrechnung," B. G. Teubner, Leipzig (1909).

3. Eddy, R. P., "Acceleration of Convergence of a Vector Sequence by Reduced Rank Extrapolation," DTNSRDC Report 81/084 (Dec 1981).

4. Knopp, K., "Theory and Application of Infinite Series," (Second English edition) Blackie & Son, London and Glasgow (1951).

5. Kline, M., "Mathematical Thought from Ancient to Modern Times," Oxford University Press (1972).

6. Wall, H. S., "Analytic Theory of Continued Fractions," D. Van Nostrand Company, New York (1948), Reprinted by Chelsea, New York (1974).

7. Baker, G. A., "Essentials of Padé Approximants," Academic Press, New York (1975).

8. Perron, O., "Die Lehre von den Kettenbrüchen," B. G. Teubner, Leipzig 2nd edition (1929), 3rd edition (1954/1957), 2nd edition reprinted by Chelsea, New York (1950).

9. Aitken, A. C., "On Bernoulli's Numerical Solution of Algebraic Equations," Proceedings of the Royal Society of Edinburgh, Series A, Vol. 46, pp. 289-305 (1926).

10. Shanks, D., "An Anology Between Transients and Mathematical Sequences and Some Nonlinear Sequence-to-Sequence Transforms Suggested by It - Part I," Naval Ordnance Laboratory Memorandum 9994, U.S. Naval Ordnance Laboratory, White Oak, MD (July 1949).

11. Shanks, D., "Non-Linear Transformations of Divergent and Slowly Convergent Sequences," Journal of Mathematics and Physics (MIT), Vol. 34, No. 1, pp. 1-42 (1955).

12. Brezinski, C., "Accélération de la Convergence en Analyse Numérique," Lecture Notes in Mathematics 584, Springer Verlag, Berlin, Heidelberg, New York (1977).

13. Brezinski, C., "Algorithmes d' Accélération de la Convergence, Etude Numeriqué," Editions Technip, Paris (1978).

14. Wynn, P., "On a Procrustean Technique for the Numerical Transformation of Slowly Convergent Sequences and Series," Proceedings of the Cambridge Philosophical Society, Vol. 52, No. 4, pp. 663-671 (1956).

15. Nörlund, N. E., "Vorlesungen über Differenzenrechnung," Verlag von Julius Springer, Berlin (1924). Reprinted by Chelsea, New York (1954).

16. Milne-Thomson, L. M., "The Calculus of Finite Differences," Macmillian and Co., London (1933).

17. Brezinski, C., "Études sur les ε - et ρ - Algorithmes," Numerische Mathematik, Vol. 17, No. 2, pp. 153-162 (1971).

18. Brezinski, C., "Conditions d'Application et de Convergence de Procédés d'Extrapolation," Numerische Mathematik, Vol. 20, No. 1, pp. 64-79 (1972).

19. Wynn, P., "On a Device for Computing the $e_m(S_n)$ Transformation," Mathematical Tables and Other Aids to Computation, Vol. 10, No. 54, pp. 91-96 (1956).

20. Gragg, W. B., Jr., "The Padé Table and Its Relation to Certain Algorithms of Numerical Analysis," SIAM Review, Vol. 14, No. 1, pp. 1-62 (1972).

21. Wynn, P., "L' ε -Algorizmo e la Tavola di Padé," Rendiconti di Mathematica e delle sue Applicazione, Roma, Vol. 20, pp. 403-408 (1961).

22. Wynn, P., "Acceleration Techniques for Iterated Vector and Matrix Problems," Mathematics of Computation, Vol. 16, No. 79, pp. 301-322 (1962).

23. Wynn, P., "Singular Rules for Certain Nonlinear Algorithms," Nordisk Tidskrift for Informationsbehandling (BIT), Vol. 3, pp. 175-195 (1963).

24. Todd, J., "The Condition of Finite Segments of the Hilbert Matrix," pp. 109-116 of Taussky, O. (Editor), "Contributions to the Solution of Systems of Linear Equations and the Determination of Eigenvalues," National Bureau of Standards, Applied Mathematics Series #39, U.S. Government Printing Office (1954).

25. Savage, I. R. and E. Lukacs, "Tables of Inverses of Finite Segments of the Hilbert Matrix," National Bureau of Standards Applied Mathematics Series 39, pp. 105-108 (1954).

26. Fettis, H. E. and J. C. Caslin, "Eigenvalues and Eigenvectors of Hilbert Matrices of Order 3 Through 10," Mathematics of Computation, Vol. 21, No. 99, pp. 431-441 (1967).

INITIAL DISTRIBUTION

Copies	
4	CNR
	1 ONR 430
	1 ONR 430C/M. Cooper
	1 ONR 432/S. Brodsky
	1 ONR 438/R. D. Cooper
1	NRL/LIB
1	USNA/LIB
3	NAVSEA
	1 SEA 03F/J. L. Schuler
	1 SEA 321/R. G. Keane
	1 SEA 996/LIB
1	NAVPGSCOL/LIB
2	NSWC/MD
	1 Dr. A. H. Van Tuyl
	1 LIB
1	NSWC/VA/LIB
12	DTIC
1	NBS/LIB
1	NASA/LEWIS RES. CENTER Dr. William F. Ford
2	CIT/DEPT. OF MATH
	1 Prof. Herbert B. Keller
	1 Prof. John Todd
1	DUKE UNIVERSITY/DEPT. OF MATH Prof. David A. Smith
1	UNIVERSITY OF KENTUCKY/ DEPT. OF MATH Prof. William B. Gragg, Jr.
1	UNIVERSITY OF MARYLAND/ DEPT. OF MATH Dr. Daniel Shanks
1	UNIVERSITY OF WISCONSIN/ DEPT. OF STAT. Prof. Grace Wahba

Copies	
1	Prof. Gene H. Golub Serra House, Serra Street Stanford, CA 94305
1	Prof. T. N. E. Greville 700 Glenview Drive Madison, WI 53716
1	Dr. Harry Polachek 11801 Rockville Pike Rockville, MD 20852
1	Dr. Feudor Theilheimer 2608 Spencer Road Chevy Chase, MD 20015
1	Prof. Richard S. Varga 7065 Arcadia Drive Parma, OH 44129
1	Mr. Thomas S. Walton 9816 Rosensteel Avenue Silver Spring, MD 20910

CENTER DISTRIBUTION

Copies	Code	Name
1	1802.1	Lugt
1	1805	Cuthill
2	1809.3	Harris
1	1826	Meals
1	184	Schot
1	184.1	Femgold
10	1843	Eddy
2	1843	Haussling
1	1843	Shoaff
1	1843	Telste
1	1843	Van Eseltine
1	1844	Dhir
1	1844	Everstine
1	1844	Gignac
1	1844	Henderson
1	1844	Schrueder

Copies	Code	Name
1	1870	Price
10	5211.1	Reports Distribution
1	522.1	Unclassified Lib (C)
1	522.2	Unclassified Lib (A)